Université du Québec
École de technologie supérieure
Département de génie de la production automatisée

GPA777 Introduction au génie logiciel

Chapitre 1

Besoins et motivations,
caractéristiques désirées
et définitions du génie logiciel

1

Introduction

- Le but du génie logiciel est de créer des produits.
- Ces produits sont des systèmes de traitement d'information.
- Ces systèmes sont réalisés sous forme de logiciels.
- Ces produits logiciels sont livrés à des clients, prêts à être utilisés.

2

Catégories de produit logiciel

- Selon [SOMM96], il existe deux grandes catégories de produit logiciel:

Produit	Commentaire
Logiciels génériques	Ce sont des systèmes autonomes vendus au grand public.
Logiciels sur mesure	Ce sont des systèmes autonomes développés selon les spécifications du client.

- Cette catégorisation n'englobe pas tous les aspects du marché (très segmenté).
- Ex: produits CAO/FAO → clientèle ciblée, les « plug-in » → systèmes partiels.

3

Quelques constatations (1)

- Malgré la simplicité de cette catégorisation, on peut tirer les constatations suivantes [PRES97]:
  - **Les logiciels sont développés et non manufacturés au sens traditionnel du terme;**
  - **les logiciels ne se dégradent pas;**
  - **la plupart des logiciels sont construits de toute pièce et non assemblés à l'aide de composants existants.**

4

Quelques constatations (2)

- La grande différence entre les produits logiciels et les autres produits traditionnels:
  - La fabrication d'un produit logiciel comporte beaucoup moins de risques et problèmes;
  - ex: fabrication des CD-ROM (ou DVD) et du manuel d'utilisation versus la fabrication d'un train d'atterrissage;
  - par contre, le coût de production d'un logiciel est surtout concentré dans la phase de développement.

GPA777 - Introduction au génie logiciel

Quelques constatations (3)

- Le processus de développement comporte des problèmes multi-factoriels qui sont difficiles à résoudre:
  - Les problèmes de gestion du personnel;
  - les problèmes de répartition des ressources (humaines et matérielles);
  - les problèmes technologiques;
  - les contraintes de réalisation imposées par le client (ou les spécifications du client);
  - etc..

Quelques constatations (4)

- L'usure ou la dégradation n'est pas un problème pour les logiciels.
- C'est la <u>désuétude</u> qui est problématique.
- La vitesse avec laquelle un logiciel est considéré comme périmé est problématique à plusieurs niveaux.

Quelques constatations (5)

- La désuétude peut augmenter les possibilités de l'incompatibilité entre sous-systèmes.
  - Ex: **Word 97** utilise les bibliothèques dynamiques (DLL) wwintl32.dll, ADVAPI32.dll, GDI32.dll, KERNEL32.dll, MSO97.dll, ole32.dll, SHELL32.dll, USER32.dll, WINSPOOL.DRV;
  - il faut s'assurer que la désuétude de ces bibliothèques n'entraîne pas de problème pour le logiciel;

Ligne de commande: dumpbin /dependents "d:\microsoft office\office\winword.exe"

Quelques constatations (6)

- La désuétude peut effacer l'avantage technologique du produit.
  - Ex: utilisation de OLE (*Object Link and Embedding*) versus COM (*Component Object Model*).
- Évidemment, la désuétude diminue l'attrait du produit auprès des clients potentiels (facteur psychologique).
  - L'incorporation de COM dans le produit est plus « exotique » que l'utilisation du « vieillot » OLE.

GPA777 - Introduction au génie logiciel

Quelques constatations (7)

- Aujourd'hui encore, les logiciels sont développés à l'aide d'outils conventionnels tels:
  - Le compilateur;
  - le dévermineur;
  - le détecteur d'erreurs en-ligne;
  - le rapporteur de performance;
  - le cadre de travail (*framework*);
  - etc..

Souvent assemblés dans un même environnement de développement (*IDE*)

Quelques constatations (8)

- Les applications ne sont pas assemblées à l'aide de composants déjà existants.
  - Pourtant l'industrie de l'électronique (et d'autres industries) ont adopté depuis longtemps cette stratégie de production.
- Une des raisons qui explique cet état de l'industrie informatique est le manque de standards dans la définition des composants logiciels.

GPA777 - Introduction au génie logiciel

Quelques constatations (9)

- Il ne faut pas croire que les produits logiciels sont différents des produits électroniques (ou autres).
- Il est tout à fait possible de créer un logiciel par l'assemblage de composants pré-fabriqués.
  - Preuve: il est possible de produire des logiciels multi-plateformes (Java, Perl et même C/C++);
  - acceptation de plus en plus grande de COM, CORBA et des JavaBeans.

Quelques constatations (10)

- La tendance de l'assemblage par composants sera amplifiée dans les années à venir.
- Ce qui accentuera l'importance de la **réutilisation** dans le développement des logiciels.
- Il y aura donc création d'un nouveau segment de l'industrie entièrement dédié à la création et à la vente des composants logiciels.

GPA777 - Introduction au génie logiciel

Caractéristiques des logiciels (1)

- Les caractéristiques désirées de tous les logiciels sont:

Caractéristique	Description
Maintenance facile	Le logiciel doit pouvoir évoluer pour s'adapter aux besoins changeants des clients.
Fiable	Le logiciel ne doit pas causer de dommage (matériel ou économique) en cas de défaillance.
Efficace	Le logiciel ne doit pas utiliser les ressources informatiques (temps CPU, mémoire, espace disque, etc.) inutilement.
Utilisable	Le logiciel doit présenter une interface conviviale et une documentation adéquate.

Caractéristiques des logiciels (2)

- L'importance de ces caractéristiques variera selon la nature du projet de développement.
  - Ex: un logiciel évoluant dans un système embarqué doit nécessairement être très efficace à cause de la limitation des ressources;
  - Ex: l'aspect esthétique et la facilité d'utilisation sont des caractéristiques primordiales d'un produit destiné au grand public.

15

Caractéristiques des logiciels (3)

- Peu importe le type de projet de développement, le respect de ces caractéristiques mènent vers un produit de qualité.
- Or, il est très difficile d'atteindre la qualité satisfaisante sans méthode de développement.
  - Imaginer le parallèle: concevoir une chaîne de montage automatisée sans planification ni méthode de travail.

16

Étude de cas (1)

- Les exemples suivants montrent combien est important l'application des méthodes et procédures systématiques et éprouvées dans le développement des logiciels.
  - La « vermine Internet »;
  - la défaillance de l'Ariane 501;
  - la complexité du transporteur de cargo C-17;
  - le lancement de la navette spatiale Columbia;
  - la désinstallation de Word.

Étude de cas (2)

- **La vermine Internet** [SOMM96]
- **Nature**: Interruption de la disponibilité et réduction de la fiabilité des systèmes à cause des failles de la sécurité.
- **Scénario**: Au mois de novembre 1988, un programme a été délibérément laissé en circulation à travers les machines VAX et SUN à l'université Cornell. Ce programme se reproduisait sans cesse et était capable d 'auto-propagation à travers l'Internet. Les machines UNIX infectées ont été surchargées par les nombreux processus créés par le programme.

Étude de cas (3)

- **La vermine Internet** [SOMM96]
- **Conséquence**: La perte de service de millier ordinateurs UNIX à travers l'Internet.
- **Cause**: Le programme profitait d'une défaillance du daemon `fingerd`. Ce dernier accepte des commandes sous forme d'une chaîne de caractères. En insérant des instructions machines après la chaîne de commande normale, il était possible d 'exécuter une sous-routine avec les privilèges de l'administrateur (*root*).

19

GPA777 - Introduction au génie logiciel

Étude de cas (4)

- **La vermine Internet** [SOMM96]
- **Cause**: Le programme profitait également d'une vulnérabilité de `sendamil`. Ce dernier est responsable de transport des courriels. Une capacité de déverminage de `sendmail` a été laissée en place. Il était possible d'envoyer des commandes vers `sendmail` afin qu'il transmet un contenu vers des destinations sans avoir à le traiter. Aussi, le programme utilise une caractéristique du système d'exploitation UNIX: les machines de confiance. Une machine de confiance (*trusted host*) permet à une autre machine d'exécuter des commandes à distance sans avoir à donner le mot de passe.

20

Étude de cas (5)

- **La vermine Internet** [SOMM96]
- **Constatations**:
  - Le programme malicieux profitait des failles de sécurité d'un ensemble de programmes (y compris le système d'exploitation lui-même);
  - l'interconnexion des machines (l'Internet) amplifie les problèmes causés par le programme;
  - un grand nombre de programmes de nature semblable mais beaucoup plus sophistiqués et malicieux ont fait leur apparition;
  - les machines PC sont devenues également vulnérables à ce type d'attaque.

Étude de cas (6)

- **Ariane 501** [BRUE00]
- **Nature**: Défaillance catastrophique du système de navigation entraînant la perte de guidage de la fusée.
- **Scénario**:
  - En juin 1996, 37 sec après le décollage, la fusée Ariane 501 (prototype de la version Ariane 5) fut détruite volontairement par une explosion;
  - les deux ordinateurs du système de navigation ont été victimes de problèmes logiciels;
  - il y a eu perte totale au niveau du guidage de la fusée;
  - la destruction de la fusée a été enclenchée automatiquement par un mécanisme d'urgence.

Étude de cas (7)

- **Ariane 501** [BRUE00]
- **Cause**:
  - L'ordinateur principal du système de navigation a connu une erreur de débordement arithmétique peu après le décollage;
  - l 'ordinateur principal entre alors en état d'exception, transfère le contrôle au second ordinateur de bord et procède à l'arrêt son opération;
  - le second ordinateur de bord a subi le même problème après une centième de seconde;
  - les deux ordinateurs cessent d'opérer l'un après l'autre;
  - la perte du système de navigation force la fusée Ariane à dériver de sa course;
  - la destruction de la fusée à été commandée.

Étude de cas (8)

- **Ariane 501** [BRUE00]
- **Cause**:
  - L'exception a été causée par un débordement arithmétique;
  - la même exception a été la cause de défaillance du second ordinateur de bord;
  - le débordement arithmétique survient lors du calcul d'alignement de la fusée par rapport aux axes de son système inertiel;
  - ces calculs sont effectués lorsque la fusée est stationnaire au sol;
  - le système de calcul devait continuer à opérer pour 50 sec après le décollage;
  - par contre, les données calculées ne seront pas utilisées après le décollage.

Étude de cas (9)

- **Ariane 501** [BRUE00]
- **Cause**:
  - Le débordement arithmétique est causé par la taille de la mémoire utilisée pour le stockage de la vitesse de déplacement horizontal;
  - au sol, ce déplacement horizontal est faible et un mot de 16bits est utilisé;
  - en vol, ce déplacement horizontal peut débordé la taille de 16bits;
  - lorsqu'il y a débordement arithmétique, l'ordinateur entre en état d'exception;
  - la gestion des exceptions consiste à exécuter une procédure de délégation et de l'arrêt de l'ordinateur;

Étude de cas (10)

- **Ariane 501** [BRUE00]
- **Constatations**:
  - Le même programme avait été utilisé sur Ariane 4 avec succès;
  - le problème de débordement arithmétique était une éventualité qui n'avait pas été détectée sur Ariane 4;
  - le programme d'alignement n'avait jamais été testé avec des vrais trajectoires;
  - le système de navigation était validé par deux équipes;
  - l'équipe système indiquait les tests à effectuer et le constructeur réalisait les tests;
  - l'équipe système n'était pas informé de la possibilité de débordement arithmétique par le constructeur du système.

Étude de cas (11)

- **Transporteur de cargo C-17** [BRUE00]
- **Nature**: Avion livré par McDonnell-Douglas avait un dépassement de budget de 500 millions de dollars.
- **Scénario**:
  - Le C-17 était développé pour être le transporteur de cargo le plus performant de son temps;
  - l'avion était livré en retard et avec un énorme déficit pour la compagnie à cause des problèmes de logiciels d'avionique;
- **Cause:**
  - Les logiciels ne donnaient pas la performance et la fiabilité désirées.

Étude de cas (12)

- **Transporteur de cargo C-17** [BRUE00]
- **Constatations**:
  - Le système est trop complexe;
  - il existe, à bord du C-17, 19 ordinateurs embarqués;
  - ces ordinateurs utilisaient 80 microprocesseurs;
  - les logiciels d'avionique sont programmés avec 6 langages de programmation différents;
  - l'incompatibilité des machines et des langages de programmation créaient une situation cauchemardesque pour les concepteurs de logiciels.

Étude de cas (13)

- **Lancement de la Columbia** [BRUE00]
- **Nature**: Le premier lancement de Columbia a été annulé à cause d'un problème non détecté pendant le développement de la navette spatiale.
- **Scénario**:
  - Le vol inaugural de la navette Columbia a été annulé lors du décompte pour le décollage;
  - aucun problème significatif des systèmes mécaniques et électroniques de la navette n'avait pu être decelé;
  - l'annulation du décollage n'était pas causée par une défaillance mécanique ou électronique.

Étude de cas (14)

- **Lancement de la Columbia** [BRUE00]
- **Cause**:
  - Le problème a été retracé à une erreur de programmation;
  - le délai de synchronisation entre les 5 ordinateurs de bord de la navette spatiale avait été modifié par inadvertance par un programmeur;
  - le délai de synchronisation entre ordinateur avait été modifié de 50ms à 80ms;
  - ce qui donnait une chance sur 67 (1/67) que tous les lancements soient annulés par erreur;
  - cette modification remontait à 2 ans précédant le lancement au cours du développement de la navette;

Étude de cas (15)

- **Lancement de la Columbia** [BRUE00]
- **Constatations**:
  - Malgré des milliers de tests effectués lors du développement de la navette spatiale, cette erreur n'avait jamais été reconnue;
  - une erreur peut être commise très longtemps avant le déploiement du système;
  - la validation du système n'est pas décidable. C'est à dire que l'on ne peut pas prouver que la validation est sans faille.

Étude de cas (16)

- **La désinstallation de Word**
- **Nature**: Lors de la désinstallation de Microsoft Word, le programme demande l'insertion du CD-ROM original dans l'unité de lecteur utilisé lors de son installation.
- **Scénario**:
  - L'installation de Word exige normalement le support d 'un lecteur CD-ROM;
  - Pour désinstaller le logiciel Word, on vous exige de placer le CD-ROM de Word dans l'unité de lecteur lequel avait été utilisé lors de l'installation;
  - Si entre-temps l 'unité de lecteur CD-ROM avait changé de nom (de F: à G: par exemple), vous ne pourrez plus désinstaller le logiciel.

Étude de cas (17)

- **La désinstallation de Word**
- **Cause**:
  - Lors de l 'installation de Word, ce dernier enregistre dans le registre de Windows le chemin source où les fichiers sont copiés;
  - le désinstallateur utilise ce chemin pour effectuer son travail;
  - de plus en plus de systèmes sont munis de disque amovible (ex: Zip Drive);
  - cela pourrait changer le nom des unités de CD-ROM.
- **Constatations**:
  - Il s'agit d'une forme de dissuasion pour contrer l'installation illégale du logiciel;
  - ce problème à retardement génère beaucoup de frustration.

Définitions (1)

- **Qu'est-ce que le génie logiciel ?**
  - **Est une activité qui implique la solution des problèmes;**
  - **Est une activité qui implique l'acquisition des connaissances;**
  - **Est une activité pilotée par des motives et des raisons;**
  - **Est une activité qui applique les principes de l'ingénierie.**

Définitions (2)

- **Qu'est-ce que le génie logiciel ?**

« Le génie logiciel est l 'application des principes de l'ingénierie à la conception des logiciels. C'est l'établissement et l'application des approches méthodiques et quantifiables au développement, à l'opération et à la maintenance des logiciels dans le but d'obtenir des systèmes logiciels économiques, fiables et efficaces dans un contexte de fonctionnement pratique. »

Définitions (3)

- **Contribution de l'ingénierie à la conception des logiciels**
  - **Les principes de l'ingénierie apportent les concepts et les outils fondamentaux du design à la production des logiciels;**
  - **Ces concepts et outils permettent le développement de produits fiables tout en rencontrant les contraintes normalement associées à des projets de réalisation.**

Définitions (4)

- La démarche méthodique de l'ingénierie facilite la gestion de la complexité en proposant des stratégies éprouvées telles:
  - La réduction par division;
  - la simplification par transformation.
- la démarche méthodique de l'ingénierie favorise l'analyse systématique des problèmes de façon à éliminer les erreurs de conception;

GPA777 - Introduction au génie logiciel

Définitions (5)

- la démarche méthodique de l'ingénierie encourage également le développement et l'utilisation des techniques dont les résultats et les effets sont prédicables et reproductibles;
- les retombées découlant de la pratique du génie dans le domaine du logiciel sont directe et tangibles:
  - le renforcement de la fiabilité des systèmes logiciels par l'identification objective des composants critiques;

Définitions (6)

- **les retombées découlant de la pratique du génie dans le domaine du logiciel sont directes et tangibles:**
  - **le renforcement de la fiabilité des systèmes logiciels par l'identification objective des composants critiques;**
  - **la réduction du coût de développement par une gestion efficace des ressources matérielles et humaines;**
  - **la réalisation de logiciels conviviaux et utilisables par l 'adoption des normes et standards de l'ingénierie.**

GPA777 - Introduction au génie logiciel

Définitions (7)

- **les retombées découlant de la pratique du génie dans le domaine du logiciel sont directes et tangibles:**
  - **l'établissement des procédures et pratiques basées sur des métriques quantifiables qui assurent la qualité des logiciels produits.**

Fin du chapitre 1

- Références:

[SOMM96] Sommerville, I., *Software Engineering*. Harlow, England : Addison-Wesley, 1996.

[PRES97] Pressman, R. S., *Software Engineering, A practitionner's approach*. New York : McGraw-Hill, 1997.

[BRUE00] Bruegge, B., Dutoit, A.H., *Object-Oriented Software Engineering*. Upper Saddle River, NJ: Prentice-Hall, 2000.